REMARKS

Rejection of claims 1-20 under judicially created doctrine of obviousness-type double patenting

The Examiner rejected claims 1-20 under judicially created doctrine of obviousness-type double patenting over claims 1, 7, 9 and 12-14 of U.S. Patent No. 6,505,344 (Blais et al.) in view of Whaley et al., "Compositional Pointer and Escape Analysis for Java Programs". Applicant traverses this double patenting rejection. The Examiner has not properly interpreted the Whaley reference as described further below. The Whaley reference does not teach or suggest for an allocation to be moved from the stack to the heap as described further below in the arguments relating to claim 1. Reconsideration is respectfully requested.

Provisional Rejection of claims 1-20 under judicially created doctrine of obviousness-type double patenting

The Examiner provisionally rejected claims 1-20 under judicially created doctrine of obviousness-type double patenting over claims 1, 7, 9 and 12-14 of copending Application No. 09/865,001 (Blais et al.) in view of Whaley et al., "Compositional Pointer and Escape Analysis for Java Programs". Applicant traverses this double patenting rejection. The Examiner has not properly interpreted the Whaley reference as described further below. The Whaley reference does not teach or suggest for an allocation to be moved from the stack to the heap as described further below in the arguments relating to claim 1. Reconsideration is respectfully requested.

Rejection of claims 13-15, 18-20 and 21-23 under 35 U.S.C. §101

The Examiner rejected claims 13-15, 18-20 and 21-23 under 35 U.S.C. §101 as being direct to non-statutory subject matter. In response, Applicant has amended these claims to recite tangible signal bearing media, thereby directing these claims to statutory subject matter under 35 U.S.C. §101.

Rejection of claims 4, 5, 10 and 21-23 under 35 U.S.C. §103(a)

The Examiner rejected claims 4, 5, 10 and 21-23 under 35 U.S.C. §103(a) as being unpatentable over Whaley et al., "Compositional Pointer and Escape Analysis for Java Programs" in view of Holzle et al. (US 6,237,141). Applicant traverses the Examiner's finding of obviousness of the claims as amended.

Claim 4

Claim 4 was amended to an independent claim by incorporating the limitations of former claim 1. The cited art does not teach or suggest claim 4. In rejecting claim 4, the Examiner cited Whaley for the teaching of "changing object allocation to the heap as more information becomes available." The examiner also quoted the sentence "Instead of being processed by the collector, the object will be implicitly collected when the method returns and the stack rolls back." This portion of Whaley, and Whaley in general does not teach or suggest "changing object allocation to the heap as more information becomes available" in combination with the other claimed features. Changing the allocation of the object from the stack to the heap as claimed above is not the same as removing it during garbage collection or initially allocating the object on the heap if it escapes.

Response to Arguments

In the "Response to Arguments", the Examiner concludes that in an iterative system "if an escape occurs the allocation should obviously be moved (changed from stack) to the heap." This statement is made based on an assumption that only exists in applicant's claims. Applicant can find no basis for this conclusion in Whaley. In fact, Whaley expressly teaches away from changing an allocation of an object from the stack to the heap.

Whaley teaches a conservative approach that assumes an object escapes if ANY of the code targeted by the object has not been analyzed. The only way an object in Whaley can allocate an object on the stack is if all the code targeted by the object has been analyzed. Thus, a decision to allocate an object on the stack in Whaley is a final decision - all information that is needed to make that decision is present, and there is no additional information that has not been analyzed that might somehow change the decision. By stating that it would be obvious based on Whaley to change the allocation from the stack to the heap, the Examiner is basically saying that it would be obvious to perform some additional analysis in Whaley that is not taught or suggested in Whaley. If an object may be allocated to the stack in Whaley, that is the end of the inquiry. Under no circumstances would the allocation of any object in Whaley ever be changed from the stack to the heap.

The claimed invention turns the basic assumption of Whaley on it's head. The claimed invention makes an allocation to the stack based on all available information, even if unanalyzed code may later indicate that the object escapes once the code is analyzed. If this aggressive assumption proves to be wrong as indicated by analyzing non-analyzed code after the assumption is made, the assumption (to allocate on the stack) may be changed to heap allocation. Thus, a primary difference between Whaley and the claimed invention is the basic assumption regarding code that has not been analyzed.

Whaley assumes the object escapes, and conservatively allocates the object on the heap. The claims, in contrast, assume the object does not escape, and aggressively allocates the object on the stack, with the understanding that this allocation may have to be changed as the unanalyzed code is analyzed.

Assuming for the sake of argument that Whaley does describe an iterative approach, Whaley only teaches allocation on the heap, unless all code targeted by the object has been analyzed, and the object does not escape. Whaley assumes "an object escapes if it is returned to an unanalyzed region of the program." (Section 1.3, first paragraph). When an object is determined to escape, it is allocated to the heap. Whaley assumes an object escapes if it references an unanalyzed piece of code, and results in allocation to the heap. Further, the process described in Whaley only analyzes each method once. (Section 1.2) Thus, when all the methods have been analyzed, an object can only then be allocated to the stack if it has been determined that it does not escape. Therefore, Whaley teaches the allocation of an object to the stack once all the code the object references has been analyzed. But, at that point, when an object is allocated to the stack, the analysis is complete, since all the methods have been analyzed. There is absolutely no teaching or suggestion in Whaley for the concept of changing the allocation of an object from the stack to the heap. This very proposition makes no sense in the context of Whaley because the goal in Whaley is to allocate on the stack if all the specific requirements for stack allocation are met. If so, hooray, we allocate on the stack. There is no situation in Whaley where any object that has been allocated on the stack would be moved to the heap FOR ANY REASON. The examiner's statement regarding how it would be obvious to change an object allocation from the stack to the heap is not supported in Whaley. To the contrary, an object in Whaley may only be allocated on the stack if certain conditions are met, and once these conditions are met, there are no known or stated ways to change the allocation to the heap. In other words, a stack allocation can only happen if the object is found to not escape any of the targeted code. If this is the case, there is no way for the method to suddenly escape, as the examiner's language

would require, which would require a change in allocation to the heap. The direction in Whaley is exactly opposite - allocation on the heap, unless specified conditions are met that allow allocation on the stack. In the claims, the allocation is made to the stack based on the available information, but may be changed to the heap if the aggressive assumption is later proven to be invalid. Because Whaley does not render obvious the changing of allocation from stack to heap, the pending claims are allowable over Whaley.

Applicants believe the Examiner has mis-characterized the cited art. If the Examiner's conclusion were correct, surely there should be a reference in the document for the change in allocation from the stack to the heap. An important feature like that would not be left to supposition. Applicant has shown that a proper reading of Whaley is that the allocation is made to the stack only after obtaining complete information for objects that do not escape the analyzed regions. (See Section 1.3, last paragraph). For these many reasons, claim 4 is allowable over Whaley, and applicant respectfully requests reconsideration of the examiner's rejection of claim 4 under 35 U.S.C. §103(a).

Claim 5

Claim 5 depends on independent claim 4 amended as described above, which is allowable for the reasons given above. As a result, this claim is allowable as depending on an allowable independent claim.

Claim 10

Claim 10 was amended to include the limitations of former claim 11.
The cited art does not teach or suggest claim 10 as amended. The rejection of former claim 11 was referenced to the rejection of claim 4. In rejecting former claim 4, the Examiner cited Whaley for the teaching of "changing object allocation to the heap as more information becomes available." The cited portion of Whaley describes allocating

on the stack if the object does not escape. The examiner also quoted the sentence "Instead of being processed by the collector, the object will be implicitly collected when the method returns and the stack rolls back." This portion of Whaley, and Whaley in general does not teach or suggest "changing object allocation to the heap as more information becomes available" in combination with the other claimed features. Changing the allocation of the object from the invocation stack frame to the heap as claimed above is not the same as removing it during garbage collection or initially allocating the object on the heap if it escapes.

As explained above with reference to claim 1, Whaley does not teach or suggest changing allocation of an object from the stack to the heap. For this reason, claim 10 is allowable, and applicant respectfully requests reconsideration of the examiner's rejection of claim 10 under 35 U.S.C. §103(a).

Claims 21-23

Claim 21 is similar in scope to claim 4 and allowable for the reasons as stated for claim 4 above.

Rejection of claims 1, 6-8, 13-15, 18 and 19-20 under 35 U.S.C. §103(a)

The Examiner rejected claims 1, 6-8, 13-15, 18 and 19-20 under 35 U.S.C. §103(a) as being obvious over Whaley et al., "Compositional Pointer and Escape Analysis for Java Programs" in view of Holzle et al. (US 6,237,141) and in further view of Choi et al., "Escape Analysis for Java". Applicant traverses the Examiner's finding of obviousness of the claims as amended.

Claim 1

Claim 1 was amended to recite the additional limitations of former claims 2 and 3. The cited art does not teach or suggest claim 1 as amended. In rejecting former claim 3, the Examiner cited Whaley for the teaching of "analyzing each class as it is loaded to determine whether the newly-loaded class affects the allocation of an object" and if so "changing object allocation to the heap." The cited portion of Whaley describes allocating on the stack if the object does not escape. The examiner also quoted the sentence "Instead of being processed by the collector, the object will be implicitly collected when the method returns and the stack rolls back." This portion of Whaley, and Whaley in general does not teach or suggest "analyzing each class as it is loaded to determine whether the newly-loaded class affects the allocation of an object" and if so "changing object allocation to the heap." Changing the allocation of the object to the heap as claimed above is not the same as removing it during garbage collection or putting it on the stack if it escapes. Examiner has mis-characterized the cited art as described above with reference to claim 4, and those arguments are incorporated here. Applicants believe the claims are now in condition for allowance. Reconsideration is respectfully requested.

Claim 6

Claim 6 has been amended to include the limitations of former claim 4. The cited art does not teach or suggest claim 6 as amended. In rejecting former claim 4, the Examiner cited Whaley for the teaching of "changing object allocation to the heap as more information becomes available." The cited portion of Whaley describes allocating on the stack if the object does not escape. The examiner also quoted the sentence "Instead of being processed by the collector, the object will be implicitly collected when the method returns and the stack rolls back." This portion of Whaley, and Whaley in general does not teach or suggest "changing object allocation to the heap as more information becomes available" in combination with the other claimed features. Changing the allocation of the object to the heap as claimed above is not the same as removing it during garbage collection or initially allocating the object on the heap if it escapes. Examiner has mis-characterized the cited art as described above with reference to claim 4, and those arguments are incorporated here. Applicants believe the claims as amended are now in condition for allowance. Reconsideration is respectfully requested.

Claim 7

Claim 7 has been amended to include the limitations of former claim 3 and 9. The cited art does not teach or suggest claim 7 as amended. In rejecting former claim 3, the Examiner cited Whaley for the teaching of "analyzing each class as it is loaded to determine whether the newly-loaded class affects the allocation of an object" and if so "changing object allocation to the heap." The cited portion of Whaley describes allocating on the stack if the object does not escape. The examiner also quoted the sentence "Instead of being processed by the collector, the object will be implicitly collected when the method returns and the stack rolls back." This portion of Whaley, and Whaley in general does not teach or suggest "analyzing each class as it is loaded to determine whether the newly-loaded class affects the allocation of an object" and if so

"changing object allocation to the heap." Changing the allocation of the object to the heap as claimed above is not the same as removing it during garbage collection or initially allocating the object on the heap if it escapes. Examiner has mis-characterized the cited art as described above with reference to claim 4, and those arguments are incorporated here. Applicants believe the claims as amended are now in condition for allowance. Reconsideration is respectfully requested.

Claims 8

Claim 8 depends on independent claim 7 amended as described above, which is allowable for the reasons given above. As a result, claim 8 is allowable as depending on an allowable independent claim.

Claim 13

Claim 13 was amended herein to recite the limitations of former claims 16, and 17. The rejection of former claim 17 was referenced to the rejection of claim 3. The cited art does not teach or suggest claim 13 as amended for the reasons stated above with regards to claim 1.

Claims 14-15

Claims 14-15 depend on independent claim 13 as described above, which is allowable for the reasons given above. As a result, these claims are allowable as depending on an allowable independent claim.

Claim 18

Claim 18 was amended herein to recite the limitations of former claim 4. The cited art does not teach or suggest claim 18 for the reasons stated above with regards to claim 4.

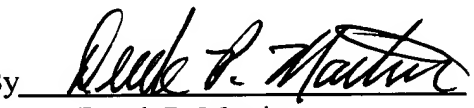
Claims 19-20

Claims 19-20 depend on independent claim 18, which is allowable for the reasons given above. As a result, these claims are allowable as depending on an allowable independent claim.

Conclusion

In summary, none of the cited prior art, either alone or in combination, teach, support, or suggest the unique combination of features in applicant's claims presently on file. Therefore, applicant respectfully asserts that all of applicant's claims are allowable. Such allowance at an early date is respectfully requested. The Examiner is invited to telephone the undersigned if this would in any way advance the prosecution of this case.

Respectfully submitted,

By 
Derek P. Martin
Reg. No. 36,595

MARTIN & ASSOCIATES, L.L.C.
P.O. Box 548
Carthage, MO 64836-0548
(417) 358-4700